Pokedex

JavaFX Homework/Tutorial

Background

Welcome to Homework 9! In this assignment you will be designing a Pokedex with JavaFX. The **Pokedex** is an electronic device designed to catalogue and provide information regarding the various species of Pokemon. The name *Pokedex* is a neologism including "Pokemon" (which itself is a *portmanteau* of "pocket" and "monster") and "index". In case you aren't familiar with the Pokemon franchise, essentially whenever you catch a pokemon its information gets automatically recorded in the pokedex.

Before we Begin…

Make note of what you need to turn in at the very end of this pdf. Spoiler alert: it is EVERYTHING you need to make your project run correctly. This includes provided code as well as any images - provided or ones you include yourself. Refer to the turn in procedure at the end of this PDF for further details.

Assignment Requirements

The Pokedex has the following requirements:

- ❏ Have a splash screen
- ❏ Be able to get to the Pokedex from the Splash Screen
- ❏ The pokedex has a list on the left side that shows the names and pictures of the Pokemon
- ❏ You can select pokemon from the list on the left side in order to view a more detailed entry.
- ❏ The pokedex has a main view with a larger picture and the Pokemon information

You are in charge of implementing all of the JavaFX Components of this assignment. You have been provided Pokemon.java and PokemonFactory.java that will provide some background implementations.

Please refer to the included video of functionality for a clearer picture of what we will be doing. If the file does not work on your computer you can also just play the video inside of Google Drive.

What is JavaFX?

JavaFX is a set of graphics and media packages. It allows us to make cross-platform GUI applications.

What is a GUI?

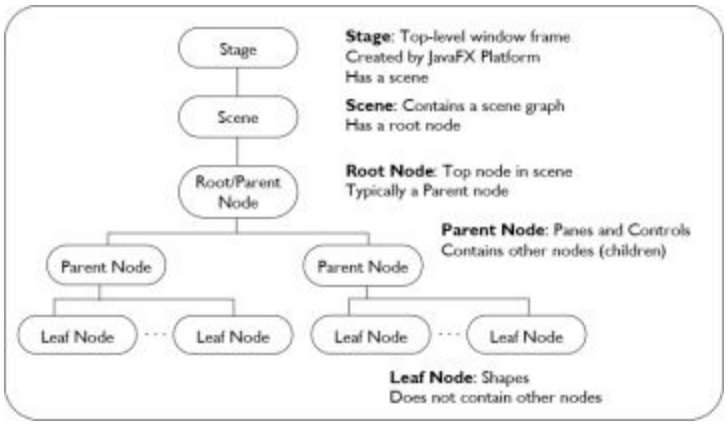
GUI stands for graphical user interface. So far this semester we have been creating programs that run through the command line. Now we will create an application that we are more experienced with - ones that use window and mouse interfaces for interaction.

A Note On Rendering

A JavaFX program's viewable elements are structured as a scene graph. The scene graph is what is actually rendered onto the screen. So if you want to add something visual to your program you need to add it to this graph.

What is a graph?

A graph in Computer Science is a series of nodes connected by edges. It is just one way to organize information. You could consider a graph a data structure. Trees are made up of **nodes** (the data) and **edges** (the connecting lines). When one node is connected to a node above it, this node is considered a **child** of its **parent** node. When a node does not have any children it is considered a **leaf node**. See the below images for examples of this vocabulary:



When you want to add something to the scene graph you will need to add it to one of the Parent Nodes as a child. We will run through a more detailed example now.

JavaFX Documentation

If you are not familiar with the API or have not been using it all semester this topic might be a little frustrating. It heavily involves making use of the javafx api in order to figure things out.

Additionally, javaFX has its own api reference:
http://docs.oracle.com/javase/8/javafx/api/toc.htm
And of course as usual you can always just use your favorite search engine to search for the class name and then "javafx api" in order to get to things faster.

And just a heads up….

There is going to be a ridiculous amount of importing required. So the API is your friend for determining packages. Just brace yourself.

Getting Started

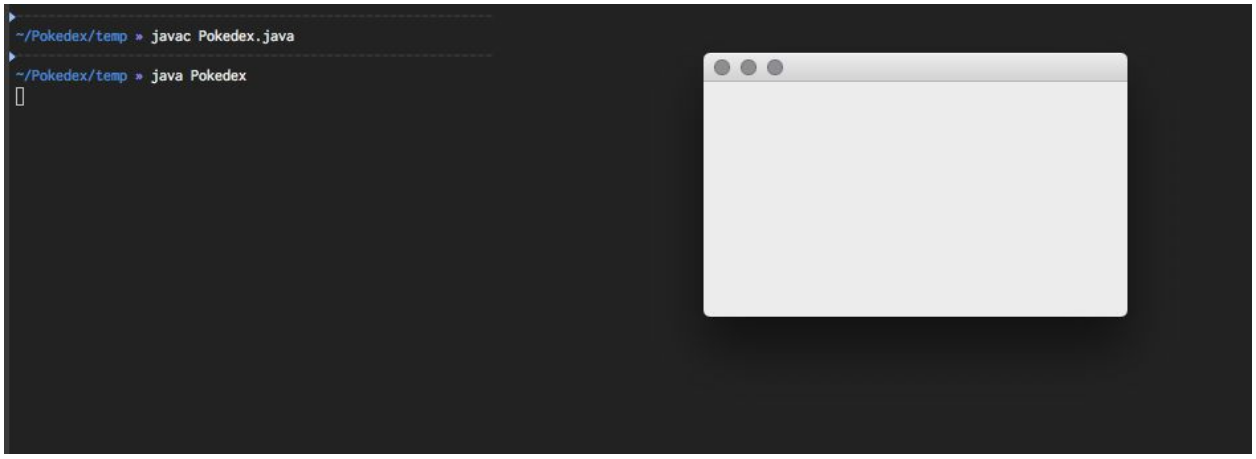
Every JavaFX application extends from `javafx.application.Application`. So to begin we need to import Application from the javafx application package and make sure our application extends from here.

```
1 import javafx.application.Application;
2
3 public class Pokedex extends Application {
4
5 }
```

Next is to make sure our program is executable. Normally we do this by creating a main method. However, with a JavaFX Application this is not necessary. What we actually need to do is implement the start method in Application. Refer to the API for this method. (https://docs.oracle.com/javase/8/javafx/api/javafx/application/Application.html)
From the API we can see that the start method takes in a `Stage`. Be sure to import `Stage` from the stage package. The Primary Stage is our primary window. In order to get it to show up we just need to call show() on it. Everything we have done so far is in the code below.

```
1  import javafx.application.Application;
2  import javafx.scene.Scene;
3  import javafx.stage.Stage;
4
5  public class Pokedex extends Application {
6
7      @Override
8      public void start(Stage primary) {
9
10
11         primary.show();
12     }
13 }
```

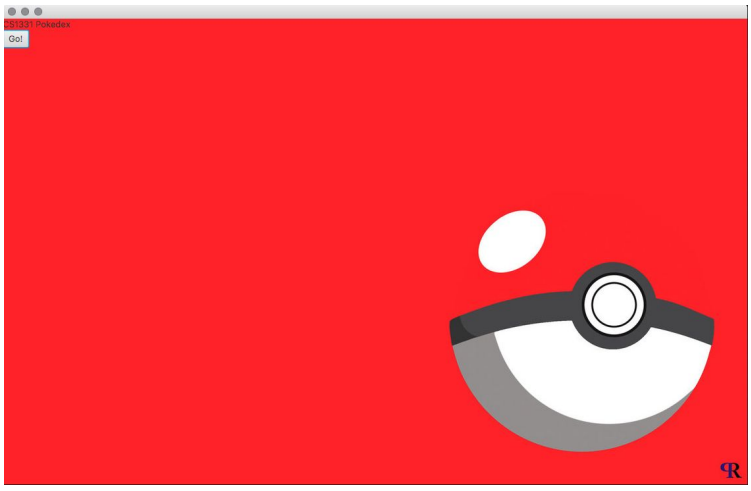
Try compiling and running your code at this point. You should see something like the image below:



Here you can see a window opens up that has nothing in it. This is our “primary stage.”
Ok we have something showing up but it isn’t very exciting. Let’s actually make things show up.

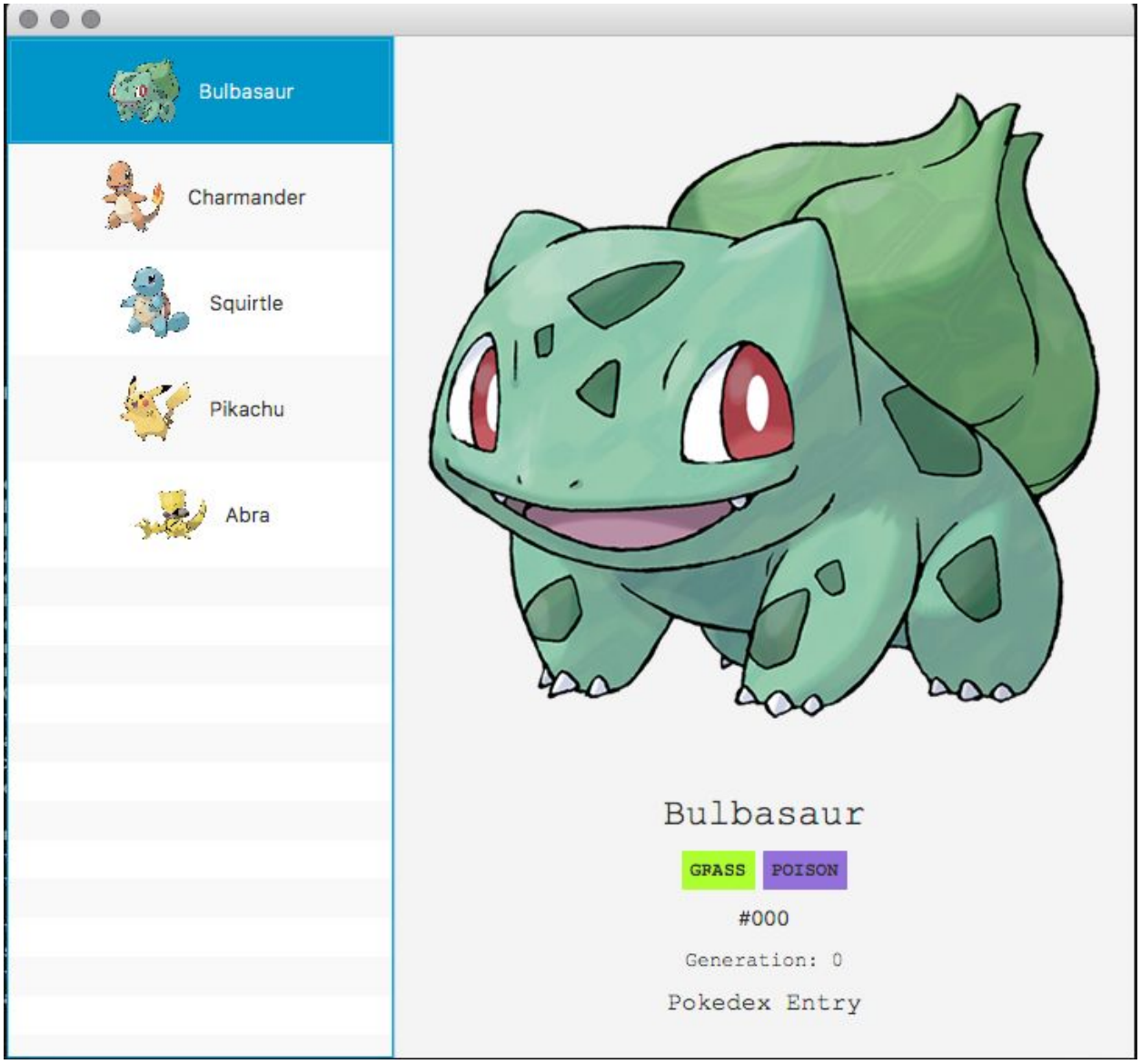
Setting a Scene

Once we have a Stage (window) we need to actually be able to put things in it. Before we can start adding images and text etc., we actually need one more thing: a Scene to add all of this stuff to. So yes it’s a very theatrical association but we have our Stage and now we need to create a Scene and then set the Scene on the stage. So the Scene we want to set up looks something like the following:



Please feel free to make this look however you please as it could look a lot cooler. However the point of this tutorial and homework is to expose you to how to implement functionality so we will not go into details about how to make it look better.

So we have the splash screen. Now what we want to do is be able to click on the button that says “Go!” and have it take us to the main functionality of our pokedex. We want our Pokedex to look like the following:



Again, there’s plenty of room for creativity here. However, we are focused on functionality. Feel free to make it look cool once you have completed the functionality portion of the assignment.

Changing the Scene

The first thing we need to determine is WHEN we want to change the scene. For this application we want to change the scene when we click on the “Go!” button. Keep this in mind.

The next thing you should notice is the layout of the Pokedex View. There is a large **center** area and a smaller **left** area. What layout should we use out of the following:

A. Vbox
B. StackPane
C. BorderPane
D. FlowPane

Determine the correct one to use. Now we need to be able to switch the scene out. Remember that we had our primary stage (window) that we were working with and we set the scene earlier to be the splash screen. Now we want to set the scene to be the actual Pokedex View.

We want to change the scene **on** the **click** of the **mouse**. Refer to the Button API to find a method that might help you do this as well as your notes from class.

When we define the functionality of the button all we need to do in order to switch the scene out is the following:

```
primaryStage.setScene(<the new scene you created that has the pokedex>);
```

That's it. When you click on the button you should see some kind of change occur in your program.

Some Notes to Help You Out

The below details are provided to prevent you from searching endlessly since there are a million different ways to do this! These should definitely help you get started.

The are on the left side should be a `ListView` (https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/ListView.html). A `ListView` is a "horizontal or vertical list of items from which the user may select, or with which the user may interact." If you look at its class hierarchy you can see that it extends from Pane just like BorderPane, StackPane, etc. So you can use it in a similar manner.

Please refer to the section "Populate a ListView" from the ListView API to figure out how to use the ArrayList that is returned by our factory method to populate the ListView!

Everything else just involves similar steps to what you did in setting up the splash screen. Details are below.

A list of classes you should get familiar with throughout this assignment:

- Button

- Label
- BorderPane
- VBox
- ImageView

Finishing the Assignment

Your next steps are to implement the rest of the JavaFX components of this application. Like mentioned previously there are a few provided files such as Pokemon.java and PokemonFactory.java that has a lot of the “backend” information completed for you so you can focus on the JavaFX portion of the assignment. However there are a few things you need to implement in Pokemon.java. The remaining requirements are outlined below:

Pokedex.java

- All of the requirements are described above.

Pokemon.java (Partially Provided)

A few methods have been outlined here for you to complete.
In particular you need to add the following abilities:

- Pokemon should have one view when they are displayed on the side panel and one view when they are being viewed in detail. For this reason we will have two different “views” for the pokemon - a side view and a detail view. You can do all of this in Pokemon or you can choose to put these in different classes. However you decide to do this is acceptable. Remember this is focused on functionality more than specific ways to do something.
- Feel free to add any getters or setters to complete your solution

Requirements for the Views

- The view in the list should at least have a small picture of the pokemon and the pokemon’s name.
- The view in the main area should have the following details
    - Picture of the Pokemon
    - Name
    - Number
    - Generation number
    - Description
    - Types

Please refer to the provided video of functionality for examples.

Feel free to add any other classes to complete your solution. Overall all requirements are based on functionality so if your solution functions the same way as the solution provided in the video you should be fine.

Aside from this please refer to “Assignment Requirements” at the beginning of this description.

Checkstyle and Javadocs

These are per the usual. Checkstyle cap is 20 points.

To Turn In

Turn in ALL FILES needed to make your project work. This includes Image files. Feel free to zip it into a compressed folder (.zip NOT a .tar-gz please and if you submit a .rar just know you will make me sad).

So your submission should include but is not limited to

- Pokemon.java
- Type.java
- Pokedex.java
- PokemonFactory.java
- All provided image files and any additional images you use in your project